JPL

Software for the Final Frontier

IEEE Automated Software Engineering Conference
26 - 29 November 2001

Peter R. Glück
NASA's Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA

JPL

Agenda

- The JPL Mission
- Types of JPL Software
- Deep Space Requirements on Vehicle Control Software
- Deep Space Application Differences
- Deep Space Vehicle Control Software
- Deep Space Life Cycle Differences
- Challenges of Deep Space
- Mitigating the Risks of Deep Space Vehicle Control Software Development
- Reducing the Cost of Deep Space Vehicle Control Software Development
- Software for the Final Frontier - A Summary

JPL

The JPL Mission

- Expand the frontiers of space by conducting challenging robotic space missions for NASA.
  - Explore our solar system
  - Expand our knowledge of the universe
  - Further our understanding of Earth from the perspective of space
  - Pave the way for human exploration
- Apply our special capabilities to technical and scientific problems of national significance.
- Our mission is what we do to implement NASA's vision.

JPL

Types of JPL Software

- Business Software
  - Oracle, web-applications, MS Office
- Engineering Software
  - Drawing, CAD, CAE, software development tools, MS Office
- Mission Software
  - Ground Data System software
    - Command generation and transmission
    - Telemetry reception, distribution, and analysis
  - Flight System software
    - Vehicle (Spacecraft) control software
    - Payload software

Deep Space Requirements on Vehicle Control Software

JPL

- Initiate and maintain control of vehicle attitude
- Deploy vehicle assets if necessary
  - Solar arrays
  - Antennae
  - Payload booms
- Monitor vehicle health for serious equipment failures
- Autonomously execute critical mission activities
  - Trajectory corrections
  - Orbit insertions
  - Entry, descent, and landing
  - Surface excursions
  - Scientific observations

Deep Space Requirements on Vehicle Control Software

JPL

- Protect vehicle assets from damage or exposure
  - Optics
  - Thermal control surfaces
  - Delicate structures
  - Deployables
- Notify operators of unrecoverable errors
- Keep the vehicle "safe" for weeks without interaction
  - Maintain power with solar arrays
  - Protect consumable resources (e.g., fuel, cryogenic coolants, switch cycles)
  - Maintain an attitude conducive to Earth communications
  - "Worry" about not hearing from Earth (did my receiver fail?)

JPL

Deep Space Application Differences

Attribute	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	***Web Browser***	***Cellular Phone***	***GPS Satellite***	***Deep Space One***
Consequence of software crash	User inconvenience	User inconvenience; possible loss of business in severe cases	Loss of service and/or replacement satellite required	Loss of science data and/or loss of mission if mission critical events are compromised
Duration of unattended operation	Minutes	Hours	Hours	Weeks
Test Method	Beta-test with volunteer users on common platforms	Test on plentiful hardware	Formal test on rare, expensive, and complex hardware	Formal test on rare, expensive, and complex hardware
Development Timescale	Months	Months to a year	Years	Years
User Interaction Timescale	Seconds	Seconds	Hours	Days
User Feedback Delay	Milliseconds	Milliseconds	Seconds	Minutes to Hours
Autonomous Operation	Low	Low	Moderate	High

JPL

Deep Space Application Differences

Attribute	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	***Web Browser***	***Cellular Phone***	***GPS Satellite***	***Deep Space One***
Interfaces	Simple: PC, keyboard/mouse, monitor	Simple: Keypad, transceiver, microphone, speaker	Complex: (usually) single payload [transceiver], vehicle control sensors and actuators [star trackers, gyros, sun sensors, thrusters, power switches, heaters, temperature sensors], communications hardware [RF transmitters & receivers, recorders, data buses], launch vehicle	Complex: **multiple** payloads [cameras, spectrometers, magnetometers, radars], vehicle control sensors and actuators [star trackers, gyros, sun sensors, thrusters, power switches, heaters, temperature sensors], communications hardware [RF transmitters & receivers, **recorders**, data buses], launch vehicle
Safety Considerations	None	None	Capability to achieve and maintain a safe state for several hours.	Capability to achieve and maintain a safe state for several **weeks**

Deep Space Vehicle Control Software Architecture

JPL

- Real-time
  - 1 to 10 Hz control of spacecraft attitude
  - Processor interrupts or polling for
    - Attitude sensor updates
    - Data bus events
    - Uplink/downlink servicing
  - Priority-based multi-task
    - High priority control and safety tasks get time when they need it
    - Low priority data processing tasks take remaining time when available
    - Task communication via interprocess communication mechanisms

Deep Space Vehicle Control Software Architecture

JPL

- Robust
  - Limited data sharing
    - Use of global variables across tasks is discouraged
    - Use of pointers across tasks is discouraged
    - Data usually passed between tasks by value
  - Protect shared information
    - Semaphores and task locks
  - Memory partitions via the operating system
    - Limits overrunning data buffers and corrupting other tasks
  - Self-monitoring
    - Must recover control quickly in the event of lock-ups or crashes
  - Limited hardware Access
    - Centralized and controlled access through a single interface

JPL

Deep Space Life Cycle Differences

Life Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Concept	Product idea and/or customer feedback	Product idea and/or customer feedback	Government or commercial objective	Scientific objective
Requirements	Developed through developer and customer discussions over days or weeks	Developed through developer and customer discussions over days or weeks	Rigorously developed to meet high-level sponsor or institutional criteria over many months. Must consider payloads, Earth-orbital environment, mission duration, frequency of contact, and operations staffing and budget.	Rigorously developed to meet scientific investigation needs over many months. Must consider **multiple payloads**, **multiple environments** [launch, cruise, planetary], **critical mission activities**, mission duration, frequency of contact, and operations staffing and budget.

JPL

Deep Space Life Cycle Differences

Life Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
High-Level Design	Developed from requirements over days or weeks. Some prototyping.	Developed from requirements weeks. Some prototyping.	Developed from requirements over months. Little prototyping. Formally reviewed by peers, management and customers prior to detailed design.	Developed from requirements over months. Little prototyping. Formally reviewed by peers, management and customers prior to detailed design.
Detailed Design	Weeks by a few individuals	Months by a few individuals to small teams	Months to years by small teams with detailed documentation and peer reviews.	Years by small to large teams with detailed documentation and peer reviews.
Implementation	Weeks by a few individuals	Months by a few individuals to small teams	Months to years by small teams with detailed documentation.	Months to years by small to large teams with detailed documentation.

JPL

Deep Space Life Cycle Differences

Life Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Unit Test	Days to weeks	Weeks	Weeks to months, usually with similar or emulated hardware	Weeks to months, usually with similar or emulated hardware
Hardware Integration	Days to weeks on mature platforms.	Weeks on (typically) mature hardware	Months on often-new prototype or engineering model hardware	Months on often-new prototype or engineering model hardware
System Test	Not applicable (standalone application)	Weeks on (typically) mature hardware	Months on the spacecraft in various environments (static testing, dynamic testing, thermal-vacuum testing)	Months on the vehicle in various environments (static testing, dynamic testing, thermal-vacuum testing)
Deployment	Via internet or disk with user interaction.	Via download or programming support equipment with technician interaction.	Installed prior to launch. Requires built-in patch and/or load capabilities through RF link with operator interaction.	Installed prior to launch. Requires built-in patch and/or load capabilities through RF link with operator interaction.

JPL

Deep Space Life Cycle Differences

Life Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth-Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Operation	Real-time interaction with user via keyboard/mouse and monitor.	Real-time interaction with user via keypad, microphone, and speaker.	Real-time interaction with user via RF link. Non-real-time execution of timed commands. Analysis of health and safety information by user. Real-time transmission of engineering and payload information.	**Delayed interaction** with user via RF link. Non-real-time execution of **sophisticated command sequences**. Often **unobservable execution of critical events** [orbit insertion, entry, descent and landing, night-side science]. **Storage of engineering information and scientific observations**. Analysis of health and safety information by user. **Playback** and transmission of scientific information to customer.

JPL

Challenges of Deep Space

- Robustness
  - Vehicle must survive hostile environments for weeks relying only on the onboard intelligence bestowed by its creators.
- Precision
  - Critical encounter events such as orbit insertion, landing, or flyby must occur at exactly the right time for exactly the right duration. The penalty for failure is **loss of mission**.
- Complexity
  - Embedded software is inherently complex since it involves interaction with the real world and the concept of time.
  - Deep Space software complexity is amplified by the many interfaces, environments, and scenarios which the software must accommodate.

JPL

Challenges of Deep Space

- Recent Mars '98 mission failures
  - Both software related
    - But both also system engineering failures
  - One due to failure to keep units straight in **ground software**
    - Mars Climate Orbiter became mistargetted and failed to achieve orbit
  - One due to failure to faithfully run planned tests on **vehicle flight software**
    - Mars Polar Lander erroneously shut down landing rockets 40 meters above the surface of Mars
  - **Both failures were preventable with proper, rigorous application of system engineering and testing principles**

Mitigating the Risks of Deep Space Vehicle Control Software Development

JPL

- Design for the worst
  - Vehicle "safe" mode is usually the most critical software
    - If we can get to safe mode, we are usually okay
  - Hardware monitors the software heartbeat
  - Advanced fault protection to
    - Detect and correct minor problems, or
    - Ensure achievement of "safe" mode
- Rigorous development practices
  - Peer reviews
  - Heritage reviews
  - Documentation at all levels
  - Coding standards enforced by acceptance reviews
  - Repeatable and documented unit tests

Mitigating the Risks of Deep Space Vehicle Control Software Development

JPL

- Test, Test, Test!
  - Test what you fly and fly what you test
  - Plan testing to verify the requirements
  - Plan testing to validate the operation
  - Test when you meet the hardware...
  - ...and when the system is completed

Reducing the Cost of Deep Space Vehicle Control Software Development

JPL

- Autocode Generation
  - Used largely in Fault Protection design but also in communication interfaces (e.g., messages, commands, and telemetry)
  - Permits specification of design in one location and at a higher, more engineer-friendly level
  - Reduces errors in translation from design to implementation
  - Reduces cost of late changes and bug fixes
  - Expanded role?

Reducing the Cost of Deep Space Vehicle Control Software Development

JPL

- Reuse
  - Emerging discipline, presently haphazard and based on similarity of mission and/or hardware
  - Formalized and intended at JPL by the Mission Data System Project
  - Still several years from application
- Improved, Integrated Design and Verification Tools
  - UML tools?
  - Model-checking of requirements, design, and implementation?
    - Forces thinking about the **properties** that the **system** must satisfy

Software for the Final Frontier
A Summary

JPL

- JPL has a mission to continue exploration in Deep Space
- Deep Space missions are unique, difficult, and complex
  - If they weren't, anyone could do it!
  - Software for Deep Space missions is also unique, difficult, and complex
- Developing Deep Space software requires rigor, discipline, and many checks and balances
  - Reviews
  - Testing
  - Documentation

Software for the Final Frontier
A Summary

JPL

- Cost benefits are being realized through the application of innovative software engineering techniques
  - Autocode generation
  - Designing for reuse
  - Advanced design and verification tools